DEEPWAVE: Deep Learning based Real-time Water Wave Simulation

Jinning Li* , Lianmin Zheng*
Group 2: 515030910592 515030910117
Department of Computer Science and Engineering,
Shanghai Jiao Tong University, Shanghai, 200240, China
lijinning@sjtu.edu.cn, mercy_zheng@sjtu.edu.cn

***Abstract*—Water surface simulation is an important task in computer animation. Today's simulation techniques are highly realistic. However, these techniques usually require extensive offline computation. It is difficult to compute the details of water wave in large phenomena. Effective real-time simulating techniques of water wave are necessary for many interactive applications such as virtual reality and games.**

We present DEEPWAVE, a novel water wave simulation technique based on deep learning and wave packet theory. Instead of solving the physics equations, we use data-driven learning-based methods to do the simulation for wave packets. There are two models available in DEEPWAVE, namely packet-wise method and area-wise method. Packet-wise method runs inference for every packet while area-wise method runs inference for every sub area patch. Experiment results show that both of them can effectively learn the motion of water wave. In addition, by training our models on better data, we can fix the artifacts produced by existing methods. Our models run very fast on modern GPU through highly optimized deep learning frameworks, so it also shows the potential to accelerate the existing CPU-based physical engine.

***Index Terms*—Water Wave Simulation, Water Packets, Neural Networks.**

I. INTRODUCTION

WAter wave simulation is an important task of computer graphics animation. Today's simulation techniques are highly realistic. However, these techniques usually require extensive offline computation. It is difficult to compute the details of water wave in large phenomena. Effective real-time simulating techniques of water wave are necessary for many interactive applications such as virtual reality and games.

In the Navier-Stokes equation, the motion of water surface waves is well described in two phases. However, the Navier-Stokes equation is quite difficult to be solved in order to simulate the water surface. To realize real-time water wave simulation, researchers usually try to apply small assumptions which will make the problem have more linear qualities. The description of water wave is also simplified into a two dimensional hight field.The linearizing of Navier-Stokes equation is applied in many simulation and description models. Some methods use addtional assumptions such as limitation of shallow water [1] and assuming static solid boundaries in [2]. Besides, there are also other methods use numercal approaches to solve the partial differential equations in order to time through the dynamics of water wave in [3].

Some more general scenarios are handled by these approaches. In the mean time, they introduce nontrivial problems, which is related to energy conservation, spatial resolution, stability, and artistic control. Besides, the waves are also approximated by some methods as Lagrangian particles [4] and Lagrangian packets [5].

However, these techniques still need to solve large quantities of equations, usually based on the simplification and linearization of in-compressible Navier-Stokes equations. These approximating techniques can significantly reduce the computational cost, but result in low quality results. This will strongly affect the user experience of some applications, especially the virtual reality application.

The wave packet model introduced in [5] shows effective performance in describing and rendering the water wave in a 2-dimensional height field. We are going to leverage the power of Lagrangian wave packets. However, instead of using mathematical methods to solve the physical equations, we use neural network to learn the mapping relation between the features among different frames.

In this paper, we present DEEPWAVE, a novel water wave simulation technique based on deep learning and wave packet theory. Instead of solving the physics equations, we use data-driven learning-based methods to do the simulation for wave packets. There are two models available in DEEPWAVE, namely packet-wise method and area-wise method. Packet-wise method runs inference for every packet while area-wise method runs inference for every sub area patch. The framework of DEEPWAVE is shown in Fig.1.

Based on the wave packet model, we describe wave packets by many parameters such as amplitude, phase, and representative wave length. The traditional method will use all these parameters to estimate the behaviours of packets, solve the physical equations, and get the states of packets in the next frame. Different from that, DEEPWAVE will extract the representative features and parameters from the parameters at current frame, transform these features into convolutional feature maps. After that, DEEPWAVE will encode these feature maps in depth through several convolutional layers. Some transpose convolutional layers are then applied to decode the information of feature maps. The output of neural network will be translated into the parameters at next frame, which determines the states of the packets in the next frame.

DEEPWAVE is a brand new technique used to simulate the water wave. Our results are competitive with existing physical

* The first two authors have equal contributions.

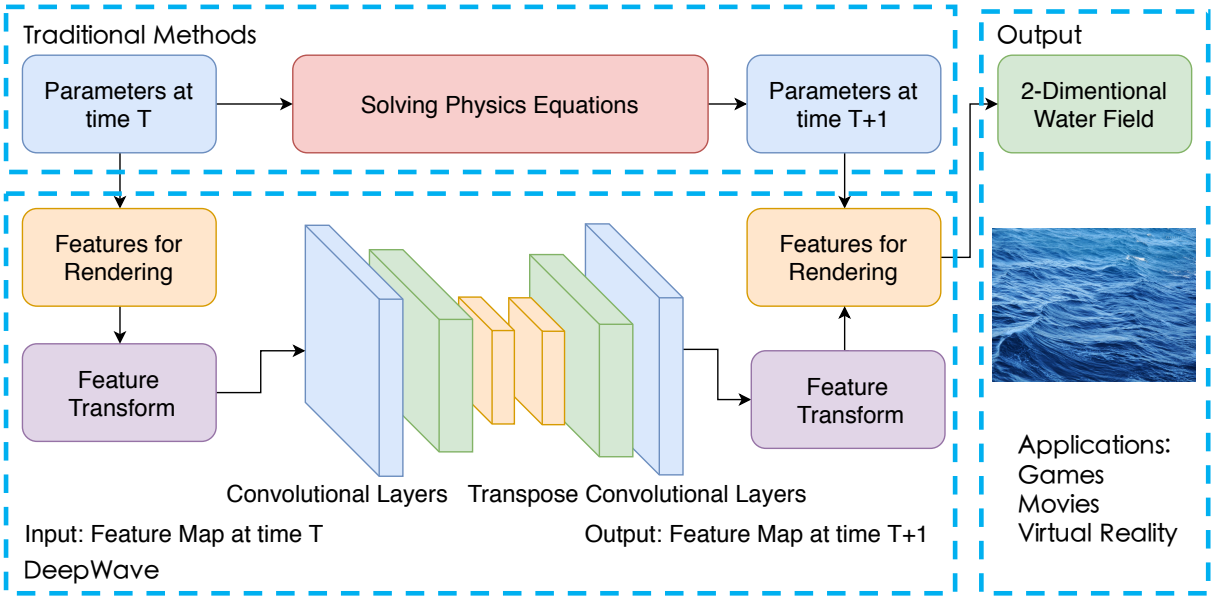

Fig. 1: Comparison between the framework of DeepWave and traditional methods

engine with high quality. This is because there are enormous physics factors can affect the movement and behaviour of water wave. The traditional techniques can only take quite limited physical phenomena into considerations, while neural network can learn inner characteristics and rules among the large quantities of data. The computational cost of convolutional neural network also allows the real-time implementation of this technique.

Our work makes the following three major contributions:

- **Data-driven** Our method is data-driven. By feeding in different training data, our models can consider different wave behaviors, such as reflection, dispersion, refraction, and so on. We can fix the artifacts in existing render engines by only improving data set. If real-world data is applied, learning model could achieve better result.
- **Realistic Result** DEEPWAVE is capable of capturing deeply inner physical laws in the training data. When data is large enough, this model will consider the physical factors that are even never used by traditional models. So, the result of this model will be much more realistic.
- **Brandnew Insight on Water Wave Simulation** Different from the traditional equation solving methods, we propose a new idea to implement real-time water wave simulation with deep neural networks. We use experiment to prove the feasibility and effectiveness of deep learning based methods. Our model can also be applied to many other kinds of fluid.

II. RELATED WORK

A. Water Wave Simulation

Since as early as 1980 [6], computer graphics researchers are interested by the animation of water waver surface. Just as what we have introduced in the introduction, the methods widely used are to apply quantities of assumptions to the Navier-Stokes equations, in order to describe the movement of the ocean. Some of them are in the form of sinusoidal waves [7], [8], [9]. All these assumptions methods sacrifice the reality of fluid movement simulation. They give us an effective computational method to solve these problem. Some similar work augmented these simple thoughts with some boundary conditions, such as breaking waves, spray, and splashes [10], [11], [12], [13], [14]. There is also a nice survey which covers the water wave simulation and ocean wave behaviours in details [15] .

In the past decade, novel methods for water wave surface simulation came into being. For example, the method in [2] will use analytical methods to solve the complex boundaries when respecting the wave behaviors such as dispersion and diffraction. These kinds of methods need large quantities of computation in advance and cannot describe the wave boundaries. Meanwhile, 2-dimensional Eulerian simulation introduced in [16] is widely used instead of some analytical Fourier based solution. The following researches are based on this method by reduce the computational errors within time [16]. There are more dispersive effects which are accurately captured in [3]. There are also some Boundary-only approaches which are able to simulate more general water wave behaviors efficiently. However, these kinds of methods will require a large amount of computation [17], [18].

The most related work is the ”Wave Particles” approach, which use its theory of particles to represent the crest of waves. This will make it easier to describe the reflection and interaction of dynamic waves. As for the control, parallelizing, and implement, it is also easier to do. Besides, when simulating some long wave and waves with high frequency, the particle model can be computational difficult. What’s worse, simulating dispersion of waves are much more difficult for this method. This is because it will introdauce a new dimention. The particle model also give as insight to transport energy

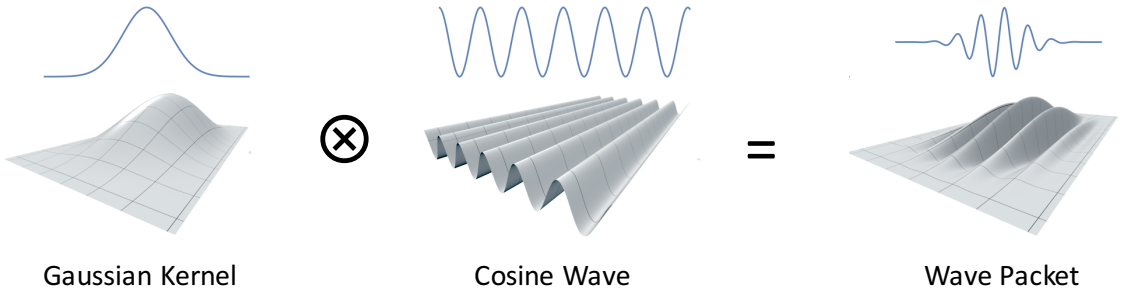

Fig. 2: The height field [5] η is the product of amplitude, kernel function and cosine wave.

of wave using the phase speed which we usually use the group speed. However, although the following method in [19] provides us with useful insights on the extending of this method, this method cannot solve the problem that how to handle wave effects like dispersion, diffraction,refraction, or boundaries reflection. Anyway, The wave particle method can inspire the following methods such as [20], which uses the background flows. And in video games simulation [21], the methods simulating time and control ability will make it an excellent model for water simulating.

However, We can use the results in many different applications, when someone chooses to simulate water waves surface. The researchers before usually used waves as guide shapes and boundary conditions [22]. Some also use it as a type of physics-based procedural texture [23], which can also be tuned in order to achieve a perfect look [24], [25]. Fully 3-dimensional simulations [14], [26], [27], [28] are also combined by water wave surface simulation.

B. Convolutional Neural Network

Some deep convolutional neural networks (DNN) [29], [30] leads to many leaps over image classification tasks in [30]. DNN will successfully integrate features in both low and high level features [31]. Beside, DNN can classify in an end-to-end multi-layer cascades. We can also enrich the feature levels by a number of stacked layers.

III. The Airy Wave Theory

A height function that varies with time, denoted as $\eta(\mathbf{x}, \mathbf{t})$, is used by the airy wave theory [32]. $\mathbf{x}$ is the position in a 2-dimensional axis. This framework is well-acknowledged to describe the state and transform of water surface. In the traditional methods, $\eta(\mathbf{x}, \mathbf{t})$ is always estimated by solving a series of simplified equations. In this paper, based on this framework, we will develop a effective learning based water wave simulation method to estimate the $\eta(x, t)$, and then visualize the result.

The surface of water wave can be viewed as an integral of many different waves and varying wavelength. This is similar to Fourier transform of the water surface.

$$\eta(x,t) = \int_{-\infty}^{\infty} a(k)cos(ks - \omega(k,h)t)dk, \tag{1}$$

where k is the wavenumber. It is inversely proportional to the wavelength. We can express this by $k = 2\pi/\lambda$, $\omega(k, h)$ is the angular frequency, it is inversely proportional to the period $T(k, h)$, which is $\omega = 2\pi/T$.

$a(k)$ is the amplitude of each wave. h is the depth of water. x is the coordinate on the selected area. t is time. There is a special form giving different features given by the equation:

$$\omega(k,h) = \sqrt{(gk + \frac{\sigma}{\rho}k^3)tanh(kh)}, \tag{2}$$

where σ is the surface tension coefficient. g is gravity. The water density is represented by ρ. the dispersion relation is a relationship between the wavenumber k and the frequency ω. the argument to the wave in Eqn.1 can be factored into $k(x - c_p(k, h)t)$ from this relation.

$$c_p(k,h) = \frac{\omega(k,h)}{k} = \sqrt{(\frac{g}{k} + \frac{\sigma}{\rho}k)tanh(kh)}. \tag{3}$$

The propagation speed in terms of a given wavelength, which is known as the phase velocity is shown in this equation. In the case of 2-dimension, with wavenumber k and a given surface area A, the energy of a water wave is:

$$E(k) = \int_A \frac{1}{2}(\rho g + \sigma k^2)(a(x))^2 dA \tag{4}$$

Wave energy travels with the group velocity c_g, which is defined as

$$c_g(k) = \frac{d\omega}{dk} \tag{5}$$

In one dimension, k, c_p and c_g are scalars. For the wavevectors magnitude $\|k\|$ n 2 dimensions, we will use the scalar k. In another words, in 2 dimensions, they are vectors $\mathbf{k}$, $\mathbf{c_p}$.

IV. Wave Packet

Wave packets [5] is an effective water wave simulating model used to describe the propagation of a series of waves. In this model, a wave packet will be represented as a collection of similar wavelengths. A larger region of space will also be covered by wave packet instead of a single wave crest. This method will let us simultaneously represent trains of long wave with single computational elements. We can let the waves interact with a dynamically changing and different environments. Also, described by Airy wave theory, we will obey the qualitative behaviors.

A. Height Field Formula

We will determine the appearance of these wave packets by the derivation of height field formula. Some properties such as reflection and dispersion will be considered so that we can give some methods later. User interactions, water depth gradients, and solid boundary are going to affect each packet as a group. Spatial coordinates should also be supported by a wave packet. We will use a kernel function, whih is denoted by $\phi(x)$, for our wave packets that acts locally in space. Its Fourier transform $\phi(k)$ also acts locally depending on wavenumber.

In [5], the author propose a formula to calculate the height field by

$$\eta(x,t) \approx \sum_{j=1}^{N} a_j \phi(x - c_g t)\cos(k_j(x - c_p t)) \tag{6}$$

This equation means that we can reasonably approximate the water surface waves by a sum of wave packets. Each is described by a kernel function. A kernel function, acts as an envelope for a single representative wave traveling at phase speed c_p and travels at group speed c_g. A spatially-varying amplitude $a(x,t) = a_j(x - c_g t)$Each packet is a attribute to each packet. Associated with packet j, a_j is a scale factor supporting spatially-constant amplitude.

The understanding of this equations is shown in Fig.2. The height field of the whole area is the sum of all the height fields built by packets. The height field of a packet is the product of amplitude a_j, kernel function$\phi(x - c_g t)$, and cosine wave $\cos(k_j(x - c_p t)$.

The equation of Eqn.6 provides a formula to calculate the water waver height wave given all the states of active packets. To understand the states of active packets, we should have a look at the packet structures and parameters.

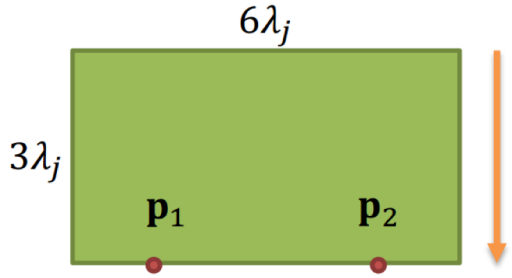

Fig. 3: The structure of a packet.

The structure of a packet is shown in Fig.3. A packet covers a square area, which is shown in green. The edge of the square is determined by the wavelength λ. There are two vertices in a packet, in convenience of packet subdivide. We will soon see how the packets subdivide. As we have mentioned, the simulation of water wave is determined by height field $\eta(x,t)$. At each time t, η can be calculated with all the parameters of packets, including positions, velocity of p_1, p_2, group velocity, phase, amplitude, and so on. The variations of these parameters realize the behaviours of water wave, such as propagation, reflection, and diffraction.

All the parameters are shown in Table.I. In the previous work, this parameter is calculated by solving physical equations. In this paper, we will use learning method to predict the variations of these parameters in the next frame and then render the results.

B. Packet Behaviours

a) Reflection: In our model, when a wave packet collides with an obstacle, it reflects. We assume that an elastic collision would perfectly preserve energy. The amplitude will be kept the same, and an inelastic collision can be simulated by decreasing the amplitude of the packet after a collision.

b) Refraction: The packet can change its group speed when it traverses through the space. It is because that c_g will change according to water depth. Then water depth is changed over the surface space. When such a situation happens, the direction of packet will be changed (i.e. refract). This process is described by Snells law [33].

c) Subdivision: We will also subdivide a packet into two when a packet deforms beyond a given threshold. It can be viewed as a procedure of adaptive refinement strategy. As can be seen, the packet is sampled from the space. Subdivision is handled in some different ways depending on whether deformation or dispersion is the source of deformation.

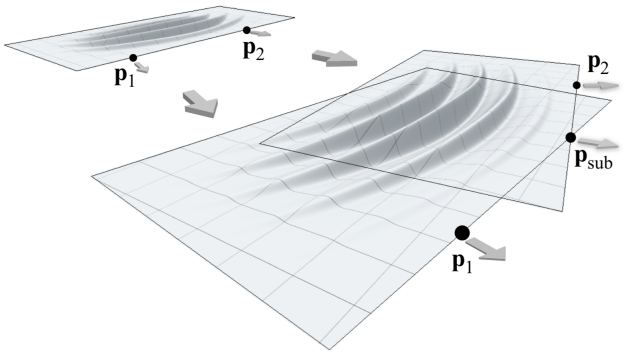

Fig. 4: The procedure of subdivision [5]. When Tangential stretching happens, the edge p_1p_2 is subdivided and the new edges p_1p_{sub} and $p_{sub}p_2$ can be seen as two new packets. This figure indicates the case after subdivision; the two new packets have been divided apart already, and the newly generated line-segments are not co-linear any more.

When the distance between two water wave packet vertices $\|p_1(t_n) - p)2(t_n)\|$ exceeds $3\lambda_j$, we can subdivide it into two packets. Or when the angle between the group velocities $c_g(p_1(t_n))$ and $c_g(p_2(t_n))$ exceeds its limit: 18 degrees.

Adding a new tracing vertex p_{sub} at the position $(p_1+p_2)/2$ and set $p_1 = p_{sub}$ for the no.1 packets and $p_2 = p_{sub}$ for the no.2. We can replace the original packet with two new ones.

Since every new packet has half the surface area of the original, every new packet will get half the energy of the original one. Finally, we can keep all constant parameters during the subdivision. Figure.4 is just a illustration of the procedure.

Name	Type	Description
E	float	wave energy flux for this packet
envelope	float	envelope size for this packet.
w0	float	angular frequency
k	float	current wavenumber.
dAmp	float	amplitude change in each timestep.
pos1,pos2,pos3	Vector2f	2D position of two vertice and a subdividing vertex.
dir1,dir2,dir3	Vector2f	current movement direction of two vertice and a subdividing vertex.
speed1,speed2,speed3	float	speed of the particle of two vertice and a subdividing vertex.
midPos	Vector2f	middle position of a packet
travelDir	Vector2f	travel direction of a whole packet
bending	float	point used for circular arc bending of the wave function inside envelope.
phase	float	phase of the representative wave inside the envelope.

TABLE I: Parameters of a packet

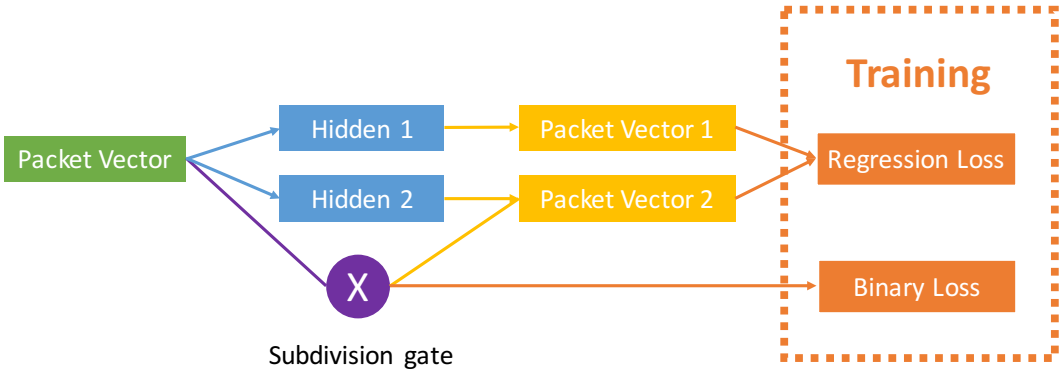

Fig. 5: Framework of packet-wise method.

V. DeepWave

A. Packet Prediction

The most tough task in water wave simulation is to calculate the height field $\eta(x,t)$ for the whole area, in different frame t. Eqn.6 provides a formula to calculate $\eta(x,t)$ using states of packets. We use $\mathbf{P_t}$ to denote the collection of packets at time t. $|\mathbf{P_t}|$ is the number of active packets. $\mathbf{p_t^i} \in \mathbf{P_t}$ represents an active packet, which is a collection of parameters shown in Table I.

We define a packet prediction task, which is a main problem solved in the following section. Given the states of all the active packet at time $t-1$, $\mathbf{P_{t-1}}$, the packet prediction task is to predict $\mathbf{P_t}$ at time t. We should notice that $|\mathbf{P_{t+1}}| \neq |\mathbf{P_y}|$ due to the subdivide and energy loss. In the case of subdivision, a packet will be subdivided to two packets, which increases the number of packets. In the case of energy loss, those packets with little energy will be wiped out, which decreases the number of packets.

The first challenge is how to encode the packet state $\mathbf{p_t^i}$. In this paper, we propose two methods, the first is to view every $\mathbf{p_t^i}$ as a packet vector. Each dimension of $\mathbf{p_t^i}$ is a parameter shown in Table I. The second method is use a feature map for every parameter, which represents the simulated region. The values of parameters of $\mathbf{p_t^i}$ are tagged in a point of a feature map. By these method the parameters are fed into a neural network in order to predict the packet in the next frame.

The challenge following is how to design the network considering the variation of packets parameters and packets number. This challenge is super vital because it determines whether the model can capture and learn the physical laws well. The most important problem is to describe the subdivision of packets, which leads to changes of packet numbers.

Based on the packet model introduced in Section IV, we propose two methods to implements the packet prediction task. The first method is a packet-wise method leveraging multilayer perceptron. This method will process every packet respectively. The framework of pair-wise model is shown in Fig.5. Input $\mathbf{p_t^i}$ is viewed as a vector, then some layers of perceptrons are designed to process the packet parameter vectors and a regression is implemented. A novel subdivision gate is designed to predict whether a packet will subdivide.

The second method is a area-wise method using Convolutional Neural Network (CNN). The area to be simulated is cut to several small patches. For each patch, we will build many 2-D feature maps. Each 2-D feature map represents a parameter in Table.I. For every packet, its parameter will be tagged on feature maps. Then, these feature will be fed in to a convolutional encoder-decoder, consisting of some convolutional layers and transposed convolutional layers. Then, the output is compared to the parameters in the next frame, loss is computed and back-propagated.

B. Packet-wise Method with Multilayer Perceptron

In this section, we will introduce a packet-wise learning method to perform packet prediction. The framework of packet-wise model is shown in 5.

The input of packet-wise method is the packet vector at time $t-1$, denoted by $\mathbf{p_t^i}$. Considering the case that the packets

will subdivide and become two packets, a gate is set to let the network learn the subdivision well. $\mathbf{p_{t-1}^i}$ is fed into two hidden layers, one of which is combined with a subdivision gate. In the case that subdivision does not happens, the subdivision gate will affect the hidden layer 2 and result in a empty output of packet vector 2. In the case that subdivide happens, the hidden layer 2 will represent the new packet at $t+1$.

During training, we use two objective function to train the network. RMSE regression loss is applied for packet vector prediction and binary classification loss is applied for subdivision prediction.

C. Area-wise Method with Convolutional Neural Network

In the Area-wise method, the area to be simulated is viewed as a picture or some feature map. Then, convolutional neural network is used to process the packet states at time $t-1$ and predict the states at time t. We use the discretization to break continuous coordinate into 2-dimensional matrix. This method is just like digital elevation model (DEM) in geometry. Assume the spatial interval is Δx, and the area is a square with side length L. Then the whole area is viewed as a feature map of size $L/\Delta x \times L/\Delta x$.

The first challenge is time consumption. The time consumption of CNN strongly depends on the size of feature map. However, sometimes the area can be extremely large. And in order to have higher accuracy, we must set the spatial interval as small as possible. which lead to a very large feature map.

So, we use some patches of size $m \times m$ to cover the area, as is shown in Fig.6. The patches are set as the input of CNN. We know that each packet $\mathbf{p_t^i}$ has its position property, which is corresponding with a point in one of the patches. We assume that there will be no two packet staying at the same position at a time. So once two packet collide in the same position, the area-wise method will ignore one randomly.

Another problem is that packet may move from patch to patch. When packet move around the boundary of a patch, it's impossible to predict that the packet will move to another patch at time $t+1$. To solve this problem, we permit overlapping between patches. There are a overlapping area between two patches, as is shown in Fig.6.

a) U-Net Architecture: The architecture of traditional convolutional encoder-decoder in the left size of Fig.7 is a good choice to deeply encode the patches and predict the values in the next frame. This architecture receive feature maps $\mathbf{x}$ with some channels. In our task, $\mathbf{x}$ is the packet parameters $\mathbf{p_{t-1}^i}$. Some convolutional layers are then used to find high level features of these packets. Then transposed convolutional layers will up-sample these high level features, which can be viewed as a prediction process.

This traditional encoder-decoder is capable to find the relationship of different packets not only spatially but also combine the information from parameters such as velocity and direction. The Abstraction ability of CNN will achieve a not bad result. However, both the low level and high level information of packets is quite vital for the packet prediction. The traditional encoder-decoder will not use the low level features to help the prediction.

In this paper, we will use the idea of skip-connection in U-Net[35] to leverage the low level features. The architecture of U-Net is shown in the right of Fig.7. The ith layer will build a skip-connection to the $(n-i)$th layer. The skip connection means a concatenation between these feature maps. This kind of combination will help the predicting process get more information from the encoder, which will result in higher precision.

Intuitively, this is because some parameter will have lower level effects on the predicted result. For example, the next position of the packet is just simple $x_{t+1} = x_t + v_g(t) + \Delta t$, which don't need high lever encoding of neural network. So skip-connection sends these features to the end without go through too much convolutional layers. On the other hand, for some parameters like phrase, energy, wavenumber, the relation between them is quite complex. To compute these parameters, the traditional method must solve many complex physical equations. So to predict these paramters, high level encoding with CNN is needed.

b) Residual U-Net: Another challenge is we need a deeper neural network to learn the complexly inner physical relation between parameters. We add some residual blocks in the middle of our model, which is capable of producing a higher level information of packets. The architecture of a residual block is shown in Fig.8. A residual blocks includes two weight layers with weight Matrix W. A skip-connection is also applied through the whole block.

Combining the advantages of U-Net in Fig.7 and residual blocks in Fig.8, we use the residual U-Net to predict the packet states in this paper. The architecture of residual U-Net is shown in Fig.6.

D. Shadow Packet

Machine Learning is a data-driven method. Our original dataset is extracted from the water wave packet simulator in [5]. We run the wave simulator offline, and get the records that how packet parameters varies with time. Then, the time series of packets states can be used train our model.

However, by observing the result of simulator in [5], we find some problems. The first problem is that water wave will have a sharply change when subdivision happens. This is because after subdivision, the initial packet will disappear and be replaced by two new packets. This change happens in a sudden, which result in a strange shake of water wave. So, we fix the training data that when subdivision happens, we don't delete the initial packet. Instead, we will copy that packet into a shadow packet, which have the same parameter as the initial one. The shadow packet will not subdivide and remains until it consumes all its energy.

By applying the shadow packets, the result tends to be more realistic, which will be introduced in Section VI. This prove that wave simulation with learning methods is capable of generalizing different kind of data. So, the higher quality the dataset has, the better performance the learning methods will achieve. This means if we can apply real-world data into this task, the simulating performance will have a big leap. This shows a bright feature of learning method.

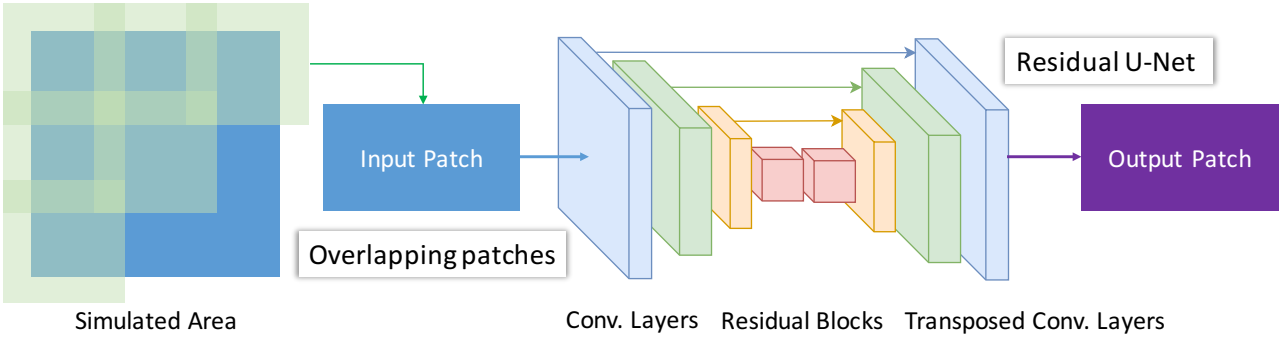

Fig. 6: Framework of Area-wise method.

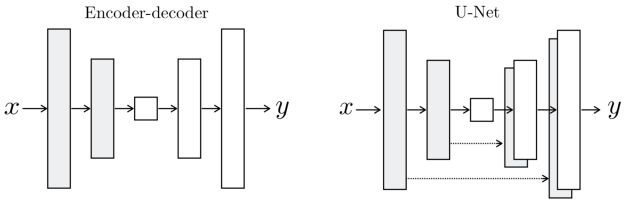

Fig. 7: Architecture of convolutional encoder-decoder and U-Net [34]

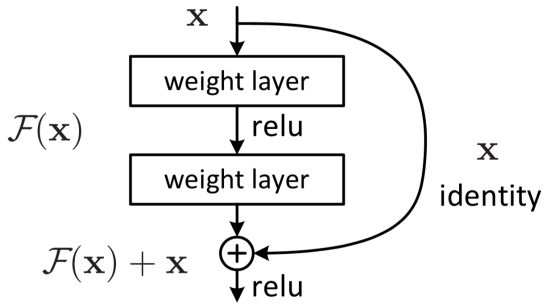

Fig. 8: Architecture of a residual block [36]

VI. Experimental Results

We compare our learned simulation against the original physical simulation engine [5] in two settings, namely simple propagation and reflection. For each settings, we collect a large data set from the original engine and use it to train our models. Then we use our models to do inference for every frame offline. Finally, the prediction results will be send to original engine for rendering. To show the advantage of data-driven method, we also conduct an experiment which fixes a artifact in rendering by the using a same model but with improved data set. Experiment results show that our models can capture most characters of the motion of water wave packet and fix artifacts with better training data.

A. Data Acquisition

We add a callback function in the original physical engine to collect data. When the original engine advects packets, we log the all the parameters (see table I) of all the packets to a text file. Table. II are some statics of the data set.

B. Simple Propagation

This is the simplest environment - a plan water surface with no obstacle, as shown in Figure 9. The original engine is supposed to work well in this setting. So the goal of this environment is to evaluate how well our model can fit the training data.

1) Packet wise: We build a multilayer perceptron as our network. We have 55 numerical features for one packet. So the input of the network is a 55-dimensional vector. Among the network, each hidden layer has 512 hidden units, and all activation layers use ReLU. We train the network using two losses as explained by Section V The optimizer is Adam with learning rate 1e-4.

2) Area wise: We build a U-Net like encoder-decoder neural network. We stack 4 previous frames as the input of the network to capture more information. So the input of the network is a 3-dimensional tensor with shape (32, 32, 4x55). The 32x32 pixel input is got by discretizing 4mx4m real number area patch from water surface. We train the network using RMSE loss and Adam optimizer.

It is not correct to compare these two methods by comparing their loss, since they use different objective functions. So we can only evaluate them by human rating. We invite five classmates (see the name list in Section IX) to rate the generated videos by physical engine, packet wise method and area wise method. The averaging rating of three methods are shown in Table III. You can visit the video in attached files.

C. Reflection

This is a more complicated setting with a rectangular obstacle, as shown in Figure 11 For packet-wise method, we add additional feature to indicate the obstacle. For area-wise method, we fill "-1" in all input channels if that pixel is occupied by a obstacle. Our model also work for this setting.

Besides, our methods show the potential to accelerate the simulation. The original engine will send a large number of packets during complicated reflection. However, it only uses CPU for the simulation. So for when the number of packets grows, its speed is bounded by simulation. However, the deep neural network frameworks are highly optimized for GPU nowadays, which brings the opportunity to make our methods faster than original cpu-based simulation.

Dataset Setting	Simple Propagation(small)	Simple Propagation (large)	Reflection (small)	Reflection (large)
Size	493 MB	4600 MB	624 MB	3800 MB
# Frame	192	437	148	235
# Water wave packet	1023105	9487213	1297313	7784332

TABLE II: Statics of Data Sets

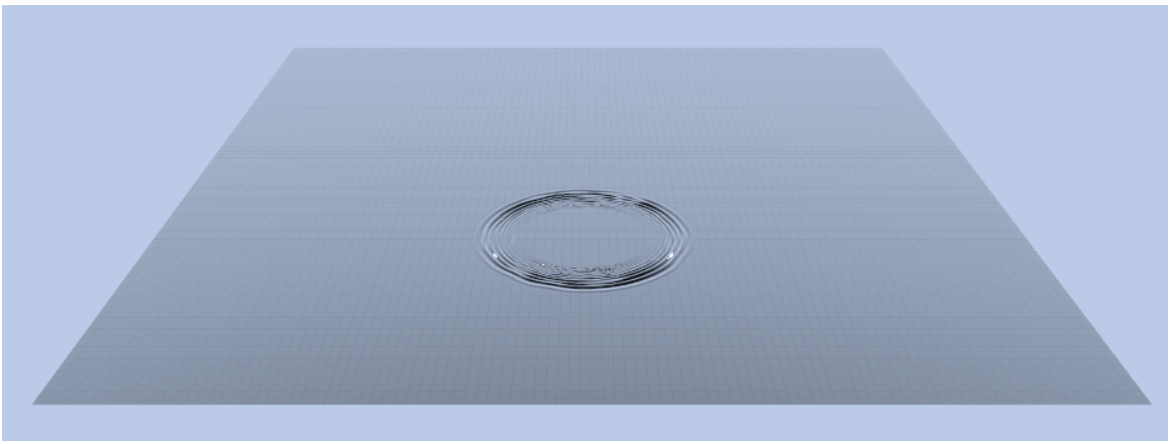

Fig. 9: Plan Surface

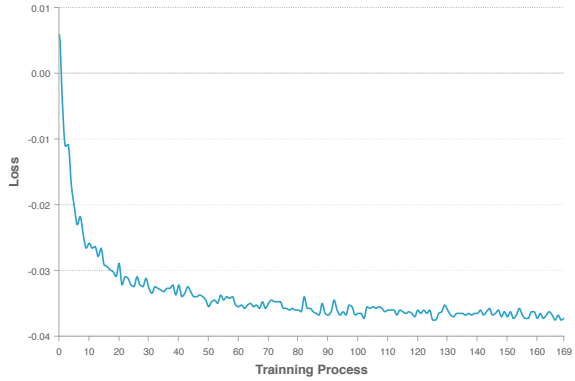

Fig. 10: Training Loss Curve of Packet-wise Method

Method	Physical Engine	Packet wise	Area wise
Average Rating	9.0	8.3	7.3

TABLE III: Human evaluation

D. Shadow Packet

In this section, we fix an artifact produced by the original engine in a data-driven way. As we know, one packet will divide itself into two packets when the angle between its two tracing vertexes is too large. However, it is hard to do this subdivision smoothly because it is a discrete process. In the original engine, we can observe some unrealistic frames when packets from a large area divide themselves simultaneously. The engine always sends newly divided packets with a small amplitude, so when the packets in a large area divide themselves simultaneously, some ”dim frames” occur during the transition. In our simple propagation setting, all packets forms some concentric circles. So all packets in a circle are symmetric and will be divide at a same time, at which the artifact occurs. To alleviate this problem, one method is to send 3 packets instead of 2 packets after subdivision. We send an extra ”shadow” packet and shade it incrementally. In this way, we can produce much smoother transition during large scale subdivision.

We manually add these ”shadow” packets to our data set. It is easy to compute the parameters for these packet since we also have subdivision information in our data set. After adding the new shadow packets, we get a better smoother data set. Then we can train a better model on this refined data set. We render the frames generated by our model and compare them against the video generated by the original engine. As shown in the attached video files, our model does not have ”dim” frames during the transition of large scale subdivision. To show the difference more significantly, we draw a curve to show the average amplitude during a subdivision, as shown in Fig. 12 and Fig.13.

VII. Limitations

There are still some limitations of our method.

- Hard to get wave packet data from real world. Since the water wave packet model is an approximate treatment in mathematics, there is no direct physical mapping from the real world to the parameters used in our feature vector. So it is not easy to collect real world data for training.
- Error explosion in long animation generation. Our system predict the next frame based on its previous prediction. So if it makes some a small error for every frame, the total error will growth exponentially. It is extremely hard for such model to generate a long animation.

VIII. Conclusion and Future Work

DeepWave provides a brand new insight on the water wave simulation task by using deep learning techniques. To simulate water wave in computer graphics animation. Today’s simulation techniques usually require extensive offline computation. It is difficult to compute the details of water wave

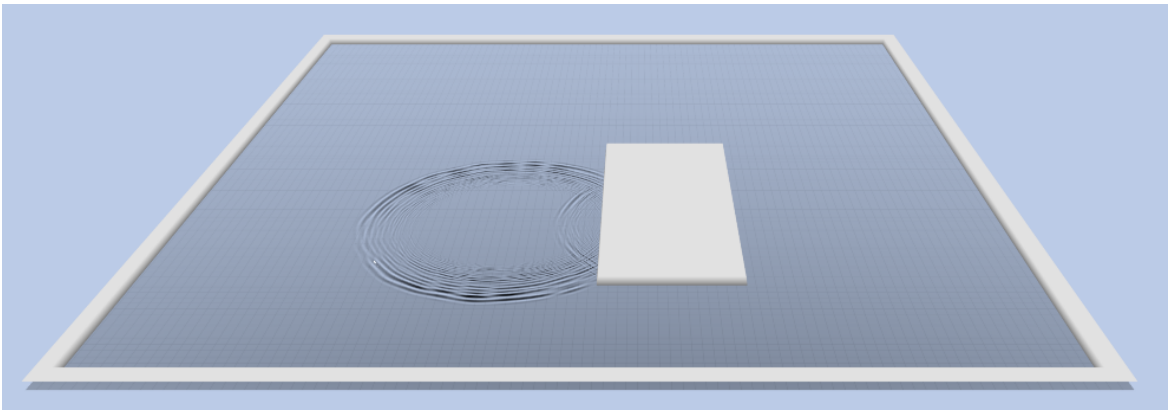

Fig. 11: Plan Surface with a Rectangle Obstacle

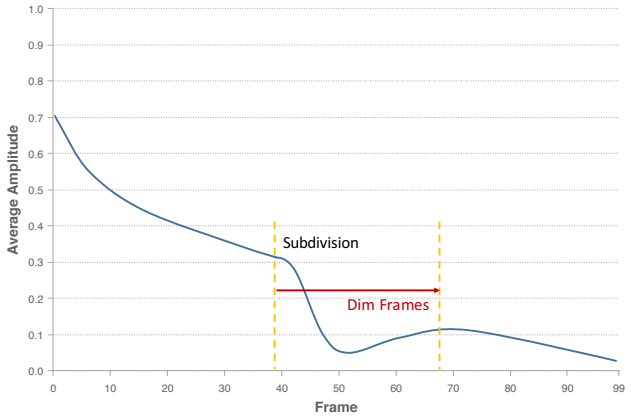

Fig. 12: Average Amplitude without shadow packets. We can observe unsmooth frames during subdivision

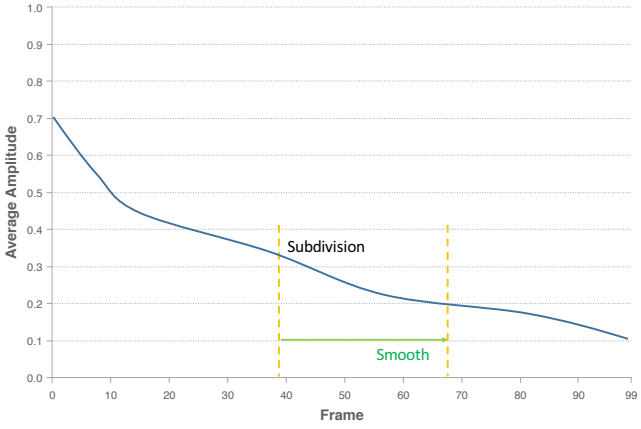

Fig. 13: Average Amplitude With shadow packets. The shadow packet pad the transition of subdivision and make it smoother.

in large phenomena. Effective real-time simulating techniques of water wave are necessary for many interactive applications such as virtual reality and games. We present DEEPWAVE, a novel water wave simulation technique based on deep learning and wave packet theory. Instead of solving the physics equations, we use data-driven learning-based methods to do the simulation for wave packets. There are two models available in DEEPWAVE, namely packet-wise method and area-wise method. Packet-wise method runs inference for every packet while area-wise method runs inference for every sub area patch. Experiment results show that both of them can effectively learn the motion of water wave. In addition, by training our models on better data, we can fix the artifacts produced by existing methods. Our models run very fast on modern GPU through highly optimized deep learning frameworks, so it also shows the potential to accelerate the existing CPU-based physical engine.

As for future work, we will train our model to better adjust the case of reflection in complicated environment. Besides, we will design more accurate architectures and loss function. We can also measure the real-world data build it as a dataset to train our model, which will result in a more realistic result.

IX. ACKNOWLEDGEMENT

Thanks Prof. Bin Sheng and teaching assistant Anum Msd for their valuable advises and help.

Thanks our evaluation rating volunteers: Lan Lou, Xiaoyu Shi, Haobing Liu, Yu Zhou, Zhou Fan.

Thanks Jinning Li's girlfriend Xiaoyu Shi and Lianmin Zheng's girlfriend Yan Li for family support.

REFERENCES

[1] M. Kass and G. Miller, "Rapid, stable fluid dynamics for computer graphics," in *ACM SIGGRAPH Computer Graphics*, vol. 24, no. 4. ACM, 1990, pp. 49–57.

[2] S. Jeschke and C. Wojtan, "Water wave animation via wavefront parameter interpolation," *ACM Transactions on Graphics (TOG)*, vol. 34, no. 3, p. 27, 2015.

[3] J. A. Canabal, D. Miraut, N. Thuerey, T. Kim, J. Portilla, and M. A. Otaduy, "Dispersion kernels for water wave simulation," *ACM Transactions on Graphics (TOG)*, vol. 35, no. 6, p. 202, 2016.

[4] C. Yuksel, D. H. House, and J. Keyser, "Wave particles," in *ACM Transactions on Graphics (TOG)*, vol. 26, no. 3. ACM, 2007, p. 99.

[5] S. Jeschke and C. Wojtan, "Water wave packets," *ACM Transactions on Graphics (TOG)*, vol. 36, no. 4, p. 103, 2017.

[6] B. Schachter, "Long crested wave models," in *Image Modeling*. Elsevier, 1981, pp. 327–341.

[7] D. Hinsinger, F. Neyret, and M.-P. Cani, "Interactive animation of ocean waves," in *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*. ACM, 2002, pp. 161–166.

[8] G. A. Mastin, P. A. Watterberg, and J. F. Mareda, "Fourier synthesis of ocean scenes," *IEEE Computer graphics and Applications*, vol. 7, no. 3, pp. 16–23, 1987.

[9] J. Tessendorf *et al.*, "Simulating ocean water," *Simulating nature: realistic and interactive techniques. SIGGRAPH*, vol. 1, no. 2, p. 5, 2001.

[10] A. Fournier and W. T. Reeves, "A simple model of ocean waves," *ACM Siggraph Computer Graphics*, vol. 20, no. 4, pp. 75–84, 1986.

[11] J.-C. Gonzato and B. Le Saëc, "A phenomenological model of coastal scenes based on physical considerations," in *Computer Animation and Simulation97*. Springer, 1997, pp. 137–148.

[12] J. F. O'brien and J. K. Hodgins, "Dynamic simulation of splashing fluids," in *Computer Animation'95., Proceedings.* IEEE, 1995, pp. 198–205.

[13] D. R. Peachey, "Modeling waves and surf," in *ACM Siggraph Computer Graphics*, vol. 20, no. 4. ACM, 1986, pp. 65–74.

[14] T. Kim, J. Tessendorf, and N. Thuerey, "Closest point turbulence for liquid surfaces," *ACM Transactions on Graphics (TOG)*, vol. 32, no. 2, p. 15, 2013.

[15] E. Darles, B. Crespin, D. Ghazanfarpour, and J.-C. Gonzato, "A survey of ocean simulation and rendering techniques in computer graphics," in *Computer Graphics Forum*, vol. 30, no. 1. Wiley Online Library, 2011, pp. 43–60.

[16] J. Tessendorf, "Interactive water surfaces," *Game Programming Gems*, vol. 4, no. 265-274, p. 8, 2004.

[17] F. Da, D. Hahn, C. Batty, C. Wojtan, and E. Grinspun, "Surface-only liquids," *ACM Transactions on Graphics (TOG)*, vol. 35, no. 4, p. 78, 2016.

[18] T. Keeler and R. Bridson, "Ocean waves animation using boundary integral equations and explicit mesh tracking," in *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. Eurographics Association, 2014, pp. 11–19.

[19] C. Yuksel, *Real-time water waves with wave particles*. Texas A&M University, 2010.

[20] M. Luboschik, H. Schumann, and H. Cords, "Particle-based labeling: Fast point-feature labeling without obscuring other visual features," *IEEE transactions on visualization and computer graphics*, vol. 14, no. 6, pp. 1237–1244, 2008.

[21] N. Tatarchuk, "Advances in real-time rendering in 3d graphics and games i," in *ACM SIGGRAPH 2009 Courses*. ACM, 2009, p. 4.

[22] M. B. Nielsen and R. Bridson, "Guide shapes for high resolution naturalistic liquid simulation," *ACM Transactions on Graphics (TOG)*, vol. 30, no. 4, p. 83, 2011.

[23] N. Chentanez and M. Müller, "Real-time simulation of large bodies of water with small scale details," in *Proceedings of the 2010 ACM SIGGRAPH/Eurographics symposium on computer animation*. Eurographics Association, 2010, pp. 197–206.

[24] C. J. Horvath, "Empirical directional wave spectra for computer graphics," in *Proceedings of the 2015 Symposium on Digital Production*. ACM, 2015, pp. 29–39.

[25] M. B. Nielsen, A. Söderström, and R. Bridson, "Synthesizing waves from animated height fields," *ACM Transactions on Graphics (TOG)*, vol. 32, no. 1, p. 2, 2013.

[26] O. Mercier, C. Beauchemin, N. Thuerey, T. Kim, and D. Nowrouzezahrai, "Surface turbulence for particle-based liquid simulations," *ACM Transactions on Graphics (TOG)*, vol. 34, no. 6, p. 202, 2015.

[27] T. Pfaff, N. Thuerey, J. Cohen, S. Tariq, and M. Gross, "Scalable fluid simulation using anisotropic turbulence particles," *ACM Transactions on Graphics (TOG)*, vol. 29, no. 6, p. 174, 2010.

[28] J. Yu, C. Wojtan, G. Turk, and C. Yap, "Explicit mesh surfaces for particle based fluids," in *Computer Graphics Forum*, vol. 31, no. 2pt4. Wiley Online Library, 2012, pp. 815–824.

[29] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.

[30] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[31] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *European conference on computer vision*. Springer, 2014, pp. 818–833.

[32] G. B. Airy, "Tides and waves," 1841.

[33] J. E. Breeding, "Velocities and refraction laws of wave groups: A verification," *Journal of Geophysical Research: Oceans*, vol. 83, no. C6, pp. 2970–2976, 1978.

[34] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," *arXiv preprint*, 2017.

[35] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[36] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.